Cómo organizar el CSS en tu proyecto

Yuri Padilha
27/10/2020

Cuando nuestro proyecto comienza a crecer y tiene un número significativo de páginas diferentes, necesitamos encontrar una forma de organizar nuestro código y comenzamos a ver un escenario muy común de:

- Código duplicado
- Conflictos de nombres de clases
- Los nombres de las clases carecen de estándar

Nosotros de **Alura** ya pensamos y repensamos cómo solucionar estos problemas a la hora de organizar nuestro **CSS**. ¡En esta publicación les hablaré un poco de este proceso!

Tomemos un componente que representa un curso de Alura, por ejemplo:

```
<!-- comienzo del componente box -->
   <a class="box" href="#">
       <img class="image" src="#">
       <h3 class="title">Curso de HTML CSS >
   </a>
```

Este HTML representa un enlace de curso de Alura. Contiene una imagen y un título.

Sin embargo, en una plataforma de cursos en línea, ¿cuál es el problema con este componente? ¡Se usa en varios lugares de la aplicación! ¿Qué significa? ¡Duplicación de

código! Aparece en la página de todas las categorias de cursos.

Por supuesto, el estilo varía un poco, pero en resumen, es el mismo código. Es decir, ¡una buena parte del CSS de este componente se puede reaprovechar!

Entonces, ¿cómo podemos reaprovecharlo? Veamos eso ahora.

Para escribir CSS tenemos que empezar creando un archivo .css. Vea que en este caso tenemos cuatro páginas, para el ejemplo, vamos hacer las páginas de los cursos de front-end y programación archivos, con dos archivos .css, uno para cada página, así nuestro código queda más organizado.

```
front-end.css

.box {}

.image {}

.title {}
```

```
programacion.css
.box {}

.image {}

.title {}
```

Y en HTML:

```
front-end.html

<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Alura | Front-End</title>
```

```
<link href="/css/front-end.css" rel="stylesheet">
</head>
<!-- todo código HTML aqui incluyendo el html de los componentes box -->
```

```
programacion.html
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Alura | Programación</title>
    <link href="/css/programacion.css" rel="stylesheet">
</head>

<!-- todo código HTML aqui incluyendo el html de los componentes box -->
```

Vale, aunque funciona, ¿cuál es el problema? ¡Los estilos se repiten! ¿Cómo reaprovecharlo entonces?

Lo que tenemos en común entre estas dos páginas es que ambas tienen el componente que llamamos **'box'**. Luego podemos aislar su css en un archivo 'box.css' y enseguida importarlo en ambas páginas.

```
box.css
.box {}

.image {}

.title {}
```

Y ahora el html

```
front-end.html

<!DOCTYPE html>
```

```
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Alura | Front-End</title>
    <link href="/css/box.css" rel="stylesheet">
    <link href="/css/front-end.css" rel="stylesheet">
</head>
<!-- todo código HTML aquí incluyendo el html de los componentes box -->
```

programacion.html

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Alura | Programación</title>
    <link href="/css/box.css" rel="stylesheet">
    <link href="/css/programacion.css" rel="stylesheet">
</head>
<!-- todo código HTML aquí incluyendo el html de los componentes box -->
```

Tenga en cuenta que también mantenemos front-end.css en front-end.html y programacion.css en programacion.html, por lo que podemos poner en esos .css el estilo particular de cada página, mientras dejamos el estilo del box en sí aislado en el box.css

¿Qué pasa si ahora quiero usar el estilo de box en otra página? ¡Simplemente importe el box.css en el html de esta otra página!

Wow, entonces ¿así lo hicieron en Alura?

En parte, sí, ¡pero aún hay más!

Hay, por ejemplo, códigos CSS que se utilizan básicamente en TODAS las páginas de la aplicación. El **reset.css**, que ayuda a eliminar la particularidad de estilo de diferentes

navegadores, es uno de ellos.

En nuestro caso, también tenemos el famoso 'container', que crea un espaciado lateral en el contenido de nuestras páginas. Para él y en casos similares, creamos un archivo base.css que se importa en toda Alura, ¡así de sencillo!

```
courses.html

<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Alura | Cursos</title>
    <link href="/css/base.css" rel="stylesheet">
    <link href="/css/box.css" rel="stylesheet">
    <link href="/css/courses.css" rel="stylesheet">
</head>
<!-- todo código HTML aqui incluyendo el html de los componentes box -->
```

Por supuesto, en el ejemplo anterior solo puse el HTML de /courses importando base.css, pero debe importarse en TODAS las páginas del proyecto.

Primero que nada, recapitulemos aquí:

1. Tenemos varias páginas en Alura. Cuando tenemos un componente que vemos que se repite, lo aislamos en un archivo aparte y lo importamos en estas páginas. Ej.: box.css al componente box.
2. Para estilos del componente que son particulares de una página, creamos un archivo css para ella y agregamos esos estilos dentro de él. Por ejemplo: courses.css a courses.html
3. Para los estilos comunes a todas las páginas, creamos un base.css que importamos a todos los HTML.
4. Estos 3 conceptos fueron extraídos del SMACSS, esta guía de buenas prácticas divide sus mandamientos en: 1. Base 2. Layout 3. Module 4. State 5. Theme

- Base es lo que llamé de base.css, que es el estilo base de todas las páginas (reset, contenedor, etc.)

- Layout es el estilo específico de cada página (front-end.css, programacion.css, etc.)
- Module es el estilo de los componentes que se repiten en varias páginas (box.css)

Espérese, ¡entremos en el state ahora!

State, más conocido en como estado, se relaciona con los estados de nuestros componentes. Piense en el siguiente componente del menú de navegación.

```
<navbar class="navbar">
    <a href="#" class="item">Home</a>
    <a href="#" class="item">Cursos</a>
    <a href="#" class="item">Blog</a>
</navbar>
```

Digamos que el usuario haga clic en el enlace **Cursos** del menú, ¿qué debería ocurrir?

1. El usuario es redirigido a la página de cursos.
2. El ítem del menú está pintado en un color diferente, generalmente un color más fuerte para indicar en qué página se encuentra el usuario.

Cuando el elemento del menú está en ese estado diferente después de pulsado, decimos que ahora está en el estado activo.

Ahora bien, ¿cómo podemos hacer con que este ítem cambie de estado? Tenga en cuenta que cambiar su color es básicamente código CSS, es decir, para cambiar el estado del ítem, ¡simplemente podemos ponerle una clase con un color diferente!

Si pulsamos en el ítem del menú, nuestro javascript debería poner la clase `.active {}` en este elemento.

¡Bam! Cambiamos su estado, ¡ahora es de otro color!

Genial, entonces tenemos este otro mandamiento de SMACSS, crea una clase para tu estado, agrégala o quítala del elemento, según necesites cambiar el estado.

Finalmente, antes de hablar del Theme que es el último mandamiento de SMACSS, hagamos un ejercicio aquí:

Dado que el menú se usa en algunas páginas, pero no en todas, ¿cómo se clasifica en SMACSS según lo que hemos visto hasta ahora?

1. Base
2. Layout
3. Module

Si eligiste C, ¡le fue bien! Es un Module (puedes llamarlo de componente). ¡Donde sea que queramos usar el menú, importamos su CSS del HTML!

Ahora, hablando sobre el Theme de SMACSS, ellos mismos en la documentación mencionan que se usa poco y, sinceramente, no lo usamos en Alura.

Esta parte es básicamente para organizar CSS cuando se trata de proyectos que pueden tener su interfaz visual modificada rápidamente por otro. Imagine sitios en los que puede cambiar una opción y cambiar al tema DARK del sitio, que cambia todo el aspecto.

Genial, gracias por leer hasta aquí, logramos organizar la estructura de nuestros archivos CSS para que no repita el código y etc., pero todavía tenemos algunos problemas que debemos enfrentar.

Si te interesa este tema, aquí en Alura tenemos capacitaciones de front-end, para principiantes y para aquellos que ya son desarrolladores web. En ellas, verás cómo programar en Javascript, HTML y CSS para construir sitios web.

ARTÍCULOS DE TECNOLOGÍA > FRONT END

un solo pago de US$49,90

- 218 cursos
- Videos y actividades 100% en Español
- Certificado de participación
- Estudia las 24 horas, los 7 días de la semana
- Foro y comunidad exclusiva para resolver tus dudas
- Acceso a todo el contenido de la plataforma por 6 meses

¡QUIERO EMPEZAR A ESTUDIAR!

Paga en moneda local en los siguientes países

ANUAL

US$79,90

un solo pago de US$79,90